REMARKS

Receipt of the Office Action of November 13, 2008 is gratefully acknowledged.

Claims 8 - 13 have been presented for examination, and these have been rejected as follows: (1) claims 8 - 13 have been rejected under 35 USC 112, second paragraph as "unclear;" and (2) claims 8 - 13 as anticipated under 35 USC 102(b) by Faist et al.

(1)

This rejection asks the question: what is a function block? The specification states on page 1, lines 16 - 19 that: "Field devices execute various functions within the process control. For special standard functions, e.g., PID-controller [proportional-integral-derivative controller] , so-called function blocks with defined communications interfaces are available." The examiner takes from this that it "...is nothing more than a processor that can be loaded into a component of a fieldbus." Actually it is more than that. Consider the attachment hereto, obtained from the internet. Note the first sentence of the Introduction which states: "[i]n engineering disciplines, especially within time driven systems, software languages are often based on function blocks. Examples are languages for programmable controllers........Many commercial software tools in the process industry also use function blocks as program organization units." Then too, "[a] generalized function block consists of input variables, output variables, through variables, internal variables, and an internal behavior description of the function block." To better define the function block behavior for the present invention, claim 8 has been amended. The amended portion of claim 8 is reproduced in annotated format showing its support in the specification:

whereby a web browser with a browser window is provided
(pg. 4, last paragraph and pg 5, second paragraph), a control

unit is provided (pg. 4, last paragraph), the control unit has access to said function block via the web browser (pg. 4 last paragraph), and with the help of the web browser the configuration and diagnosis information (pg. 3, 7th paragraph) of the other function blocks which are connected to said function block are available in a general descriptive language and edited and changeable with the help of the web browser (pg. 5, first paragraph).

With the noted amendment to claim 8, the rejection under 35 USC 112, second paragraph is believed to be overcome

(2)

The noted amendment to claim 8 is also believed to distinguish the invention over Faist et al. In his rejection, the examiner discusses the application of Faist et al against claim 8. What has been added to claim 8 is not found in the examiner's commentary. The paragraphs referred to by the examiner do not, it is respectfully submitted, teach connecting function blocks so that changes can be made. That is now recited in claim 8. It must be noted that web page templates are not considered function blocks, as function block is understood in this application and the enclosure hereto. It is not seen, therefore, that Faist et al can render claims 8, and those claims which depend therefrom unpatentable.

In view of the foregoing, reconsideration and re-examination are respectfully requested and claims 8 - 13 found allowable.

Respectfully submitted,

BACON & THOMAS, PLLC

Date: February 13, 2009

Felix J. D'Ambrosio
Attorney for Applicant
Registration Number 25,721

Customer Number *23364*
BACON & THOMAS, PLLC
625 Slaters Lane, Fourth Floor
Alexandria, Virginia 22314
Telephone: (703) 683-0500
Facsimile: (703) 683-1080
S:\Producer\fjd\CLIENTS\Endress+Hauser Holding GmbH\DASI3006-PS0022\Response Feb 13 2009.wpd

Function Blocks

Introduction

(by Torsten Heverhagen, Robert Hirschfeld, Rudolf Tracht)

In engineering disciplines, especially within time driven systems, software languages are often based on function blocks. Examples are languages for programmable controllers [IEC 61131] or simulation environments like Matlab/Simulink [Simulink]. Many commercial software tools in the process industry also use function blocks as program organization units. Despite minor differences the concept of function blocks is the same in all considered function block oriented languages.

A generalized function block consists of input variables, output variables, through variables, internal variables, and an internal behavior description of the function block. Input variables can only be written from outside of an FB. From inside they can only be read. Output variables can be read and written from inside of an FB and only be read from outside. Through variables are special shared variables. If through variables of different FB instances are connected, they do all access the variable connected to the first input of the chain. Through variables are defined in [IEC 61131]. They are often called In-Out-variables. If their datatype matches, output variables can be connected to input variables by a connector. This is similar to a connection of ports with matching protocols.
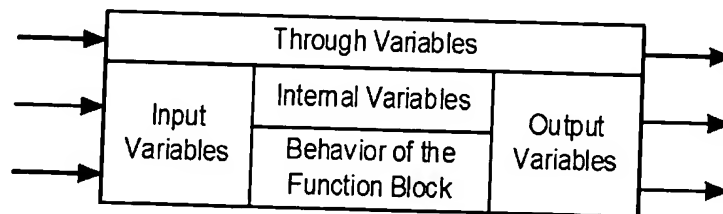

General Function Block Model

Unlike simple functions, FBs have internal state information that persist the execution of FB instances. The internal behavior can be driven by continuous [Simulink] or discrete time [IEC 61131], or can be event driven [IEC 61499]. Common to all FBs is that their interface variables (input, output, and through) continuously provide data values. Communication with other FBs can only be done by assignments of data values to interface variables. This communication model is contrary to the one of object orientation, where objects communicate by message exchange.

Interface variables of FBs cannot be compared to UML attributes. One of the reasons for this is that in object orientation it is not possible to define an attribute, which can be read and written from outside of the owning class and only be read from inside. This is required for the definition of input variables. Another reason is, that attributes of one class cannot be connected to attributes of another class by the means of connectors.

The best counterpart of interface variables in the UML are ports. Both concepts provide export interfaces, import interfaces and encapsulate the internal behavior of FBs and classes respectively. An example of a declaration of interface variables is given in the following figure:
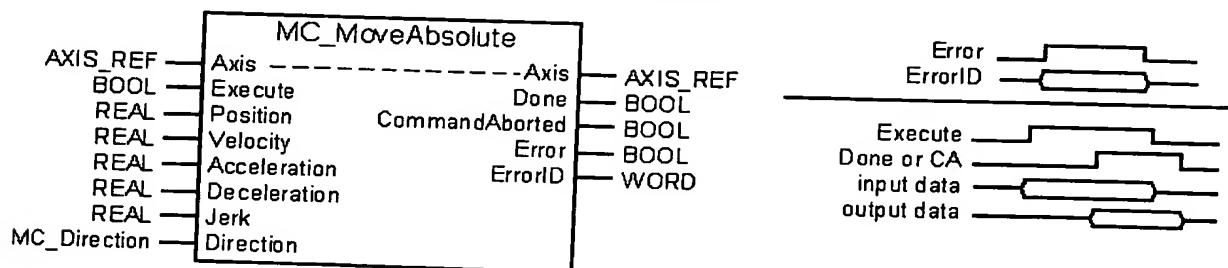
Function block MC_MoveAbsolute and timing diagrams describing its FB-protocol

3

2/13/2009 5:00 PM

The FB type MC_MoveAbsolute is taken from [PLCopen]. It is a standardized FB type responsible for the control of a motion of a motor axis to an absolute position. The notation is conformant to [IEC 61131-3]. Interface variables are denoted with input pins (left side) for input variables and output pins (right side) for output variables. Through variables are marked with dashed lines. Variable names are placed inside of the function block and datatypes outside. At the right side of this figure three timing diagrams are shown. The upper one shows that if an error occurs an ErrorID is given. Errors can occur even when the axis is not in motion. The middle timing diagram explains that after an execution of a motion is requested, the motion can be completed (signaled in Done) or aborted (signaled in CommandAborted (CA)). The so called input data is given in the set of input variables Position, Velocity, Acceleration, Deceleration, Jerk, and Direction. It is only valid, during Execute is being set. We define the so called output data as the set of output variables Done, CommandAborted, Error, and ErrorID. Our definition of input data and output data is only to get more convenient notions for the later explanations in this paper. The information given in Axis is always valid. With this information several FBs may work on the same axis. For example, an execution of a motion can be aborted by another FB. That's why CommandAborted can raise when Execute is true.

In MC_MoveAbsolute some Boolean variables are used to signal events. This is a common tech-nique for defining FB-protocols. The information given in [PLCopen] can also be used to derive a protocol statechart or a protocol automaton, which can be used for verification issues [4]. But for FBs it is not necessary to have such kind of triggering variables:
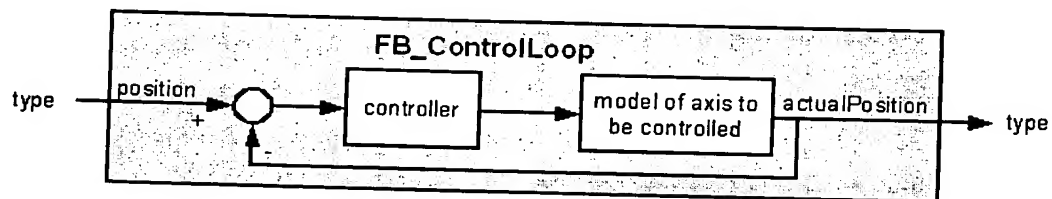
Function block FB_ControlLoop

FB_ControlLoop contains a simplified model of a control loop, which could work inside MC_MoveAbsolute. The input variable position is connected to a summation block (denoted as a circle). The second input of the summation block is actualPosition. The output of the summation block is the difference between position and actualPosition. This output is connected to the input of the controller block. The internal behavior of the controller is described with differential equations. This is the same for the block called model of the axis to be controlled. The output of this block is also the output of the overall block FB_ControlLoop. FB-diagrams like in FB_ControlLoop are used in control theory and in simulation tools like Simulink. We would like to emphasize that our generalized function block model also works with such languages. Input, output, and through variables as well as internal state information can be observed. The type of inputs and outputs is tool dependent. For pure mathematics it is of course the set of real numbers.

An actual development in the field of function block oriented languages is [IEC 61499]. In this standard the concept of function blocks is extended with event inputs and event outputs. This should ease the application of function blocks to event driven systems. All data inputs and data outputs must be explicitly connected to an event input or event output respectively. An example is shown in the next figure.
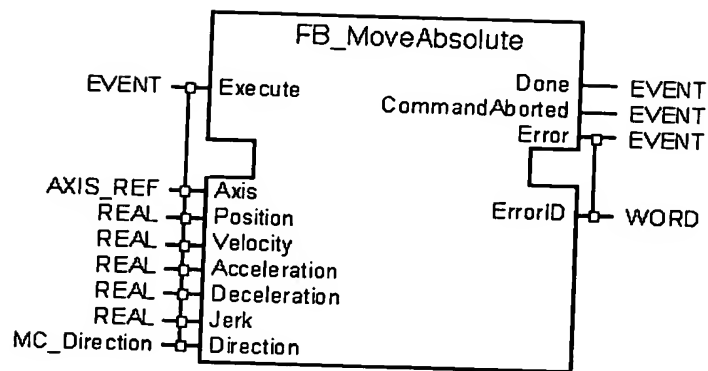
Example for an FB type conformant to IEC 61499. The FB is separated into an upper and a lower part. At the upper part the event inputs and outputs are drawn. The data inputs and outputs are drawn at the lower part. Because FBs of IEC 61499 don't have through variables, Axis is modeled as an input variable and connected to Execute like all other data inputs.

The type EVENT can be seen as an abstraction of the Boolean type. Events are stored into Boolean Event Input (EI) and Event Output (EO) variables. Compared to FBs of IEC 61131-3 the interface description of IEC 61499 has richer syntax and semantics. The generalized FB-model can be applied to IEC 61499, if event inputs are treated as simple Boolean variables.

↑ back to functionblocks.org